PARTITION CREATING METHOD AND DELETING METHOD

TECHNICAL FIELD

The present invention relates to a management technology of a storage device, particularly to a technology of managing creation and deletion of a partition of a hard disk device.

BACKGROUND ART

In a storage device, particularly in a hard disk, setting partitions makes it possible to use physically one hard disk as if it were separate hard disks. It is therefore actual practice to divide a hard disk having a large capacity into a plurality of partitions, for facilitating management of the disk or enabling booting of a plurality of OSs by switching one OS to another. Further, the partitions are also used for the purpose of preventing a decrease in a processing speed, which is caused by an occurrence of so-called fragmentation, by recording files to be frequently opened for reading and writing and system files and the like to be altered to a less degree in separate partitions.

When a user conducts partition alteration, conventionally, the user uses a utility program or the like to manually add or delete partition(s). That is, when the user adds a partition, the user takes account of a necessary partition size and a securable region in a hard disk to determine the position of partition to be created.

DISCLOSURE OF THE INVENTION

Meanwhile, the partition alteration has not been performed so

frequently. The alteration has been limited, for example, to the case of adding a hard disk, the case of changing operation systems, and the like.

However, there are generated demands for adding and deleting a partition as required thanks to enlargement of a hard disk capacity, diversification of operation systems, a necessity to handle files having various properties.

In such a case, when the addition and deletion of the partitions are disorderedly repeated, the hard disk is fragmented into minute partitions, which may lower the utilization efficiency of a hard disk device.

It is an object of the present invention to provide a technology for improving the utilization efficiency of a storage device.

For achieving the above subject, a partition creating method of creating a partition in a storage device, provided by the present invention, comprises limiting the size of a partition to be created to a size of m to the n-th power where m and n are natural numbers, and disposing the partition to be created in a position aligned by the size of the partition.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram showing a constitution of one embodiment of the partition management system according to the present invention.

Fig. 2 is a view for explaining partitions as divisional regions of a hard disk in this embodiment.

Fig. 3 is a view showing one example of a structure of a partition management table 103.

Fig. 4 is a flowchart for explaining processing in creating the

partition.

Figs. 5A to 5D are views for explaining a processing example in creating the partition.

Figs. 6A and 6B are views for explaining processing in creating the partition in a final undefined region.

Fig. 7 is a flowchart for explaining processing in deleting the partition.

Figs. 8A to 8G are views for explaining a processing example in deleting the partition.

BEST MODE FOR EMBODYING THE INVENTION

Embodiments of the present invention will be explained in detail with reference to the drawings.

Fig. 1 is a block diagram showing a constitution of one embodiment of a partition management system according to the present invention. In this drawing, a partition management system 100 includes: an interface unit 101 for receiving a program under execution, a request for adding a partition and a request for deleting a partition from an operator or the like and sending results of processing the same back thereto; a partition management execution unit 102 for executing addition and deletion of a partition in a mounted storage device and performing creation and renewal of a partition management table; and a partition management table 103. In this embodiment, a hard disk 110 as one example of the storage device is mounted in the partition management system.

The partition management system 100 includes: a central

processing unit (CPU), a main storage device, a reading device for reading data from a portable storage medium such as a CD-ROM or DVD-ROM, an input device such as a keyboard, a mouse and a controller, a display device such as a display, and an interface for controlling transmission and reception of data among the above-described constituent components. The partition management system 100 can be constituted on a generally configured information processing apparatus capable of internally or externally mounting the storage device such as a hard disk, for example, a personal computer, a server computer or an entertainment apparatus.

Further, a program for allowing the information processing apparatus to execute the processing of the partition management system 100 can be included, for example, as part of utility software, a hard disk driver, a library for development, or the like. Such a program can be recorded in a storage medium such as a CD-ROM and a DVD-ROM and distributed. Alternatively, The program can be distributed through a communication line.

In this embodiment, for specifying a sector as a unit of a recording region in the hard disk, it is determined that a logical block address (LBA) starting from 0 is used. This address makes it possible to identify any sector on the hard disk. Naturally, the present invention is not limited thereto, and there may be employed a constitution in which any sector on the hard disk is specified by designating, for example, a cylinder number, a head number and a sector number. The logical block address will be referred to as a sector number for convenience.

Fig. 2 is a view for explaining partitions as divisional regions of

the hard disk in this embodiment.

In the present invention, a partition created by the operator, the program under execution, or the like so that it permits reading/writing of data is referred to as a defined region (partition), and a storage region in which no partition is created yet or a partition created once is deleted is referred to as an undefined region. The above undefined region will be sometimes referred to as an undefined partition for convenience.

In this drawing, four regions (110a, 110b, 110c and 110d) are formed on the hard disk 110. Of these, the regions 110a and 110c are defined partitions that permit reading/writing of data. The region 110b present between the regions 110a and 110c is an undefined region. The region 110d following the region 110c is an undefined region, which is assumed to continue to the end of the hard disk 110. That is, this drawing shows a state where the partition 110a and the partition 110c are created on the hard disk in an initial state.

Of the above undefined regions, an undefined region followed by a defined region or another undefined region like the region 110b is referred to as an "empty undefined region", and an undefined region that is the last region of the hard disk like the region 110d is referred to as a "final undefined region".

Each region has a header portion having a predetermined size on a head thereof. Each header portion is to pre-record, for example, information to the effect that it is a head of the partition, a flag for identifying whether the partition (region) is already defined or undefined, and a size of the partition. The size of the partition can be represented, for example, by the number of sectors and the number of bytes.

Alternatively, an ending sector number may be recorded to indirectly represent the size of the partition.

However, the information recorded in the header portion is not limited to the above-described ones. For example, the information to the effect that the header is the head of the partition may be replaced with a leading sector number of the partition. Further, an identifier for identifying the partition, for example, a partition number may be recorded in the header portion. Furthermore, when the partition is, for example, undefined, the partition size may be set at 0 without using the above flag as the information for identifying whether the partition is already defined or undefined.

An operation of the partition management system 100 in this embodiment will be explained below. The operation of the partition management system 100 is divided into processing of creating the partition management table 103, processing of adding a partition and processing of deleting the partition.

First, the processing of creating the partition management table 103 will be explained.

The partition management table 103 is a table that is created on a main storage device, or the like, by the partition management execution unit 102 for managing information concerning the partitions on the hard disk. Fig. 3 is a view showing one example of structure of the partition management table 103.

In this drawing, the partition management table 103 has identifiers 1031, starting sector numbers 1032, sizes 1033 and statuses 1034.

The identifiers 1031 are used for identifying the partitions, and this example uses numbers in a descending order. The statuses 1034 are information for identifying whether the respective regions are already defined or undefined.

The partition management execution unit 102 creates the partition management table 103 when the partition management system 100 is activated, when the hard disk device is mounted on the storage device, or the like.

Specifically, the partition management execution unit 102 refers to the header portions recorded in the hard disk 110, acquires the leading sectors, the sizes and the information for identifying whether the region is defined or undefined with regard to each partition (including the undefined regions), and records the same in the partition management table 103.

The above management table may be created on the hard disk, for example, a managing region of the hard disk in advance. In this case, the partition management execution unit 102 can manage the partitions by referring to this region.

The processing in adding the partitions will be explained below.

In this embodiment, the size of the partition to be added is assumed to be 2 to n-th power times as large as a base unit. The base unit has an arbitrary size, and can be determined to be 1 sector (typically 512 bytes), 1 cluster, 1 kilobyte, 1 megabyte, 100 megabytes, 1 gigabyte and the like. However, this embodiment will be explained on the assumption that the base unit is one sector.

The size of the partition to be added will be any one of 1, 2, 4, 8,

16, 32, ..., 1024, ... sectors, which are 2 to the n-th power times as large as one sector. Where no particular confusion occurs in the following explanation, sizes are represented without the unit "sector". Further, the size of the partition to be added is not limited to the size that is 2 to the n-th power times as large, and can be a size that is any natural number to the n-th power times as large, for example, 3 to the n-th power, 4 to the n-th power times as large.

Even when a substantially necessary partition size is, for example, 3, therefore, the partition size which this system is requested to add is 4, which is 2 to the second power ($2^1<3\leq2^2$). Similarly, when 1000 is necessary, 1024 (2^{10}) becomes a size to be added by request. Naturally, this judgment may be made with a program or the like making such a request, while there may be employed a constitution in which the interface unit 101 is imparted with a judgment function, and for example, when the interface unit 101 receives a request to add a partition having a size of 5, a partition having a size of 8 as a minimum additional size sufficient for satisfying the request is added. For simplicity, this embodiment will be explained on the assumption that the size 2 to the n-th power times as large is requested of this system.

In this embodiment, further, a position where the partition can be located when created, that is, the position that can be set as a starting position of the partition is limited to a position (sector number) where the partition size is aligned.

Specifically, if a requested partition has a size of 256 (2^8) sectors, the location where this partition can be located is limited to regions with sector numbers 0, 256, 512, 768... (m times 256) as starting positions,

where the partition size is aligned. Further, when the size of a partition is, for example, 1 as 2 to 0-th power, the partition can be positioned at any sector number.

On the premise of the above regulations, the interface unit 101 requests the partition management execution unit 102 to execute the addition of the partition upon receipt of a request to add the partition and the size of the partition to be added. The processing of the partition management execution unit 102 having received this request will be explained with reference to the flowchart of Fig. 4.

First, the partition management execution unit 102 refers to the partition management table 103, and checks whether or not an empty undefined region equal to the requested size exists (S101).

As a result, when the empty region exists, the partition management execution unit 102 creates a partition at the requested size in the empty region (S106), and reports the creation of the partition to a requester through the interface unit 101. Further, the partition management execution unit 102 updates the partition management table 103 on the basis of the location and the size of the created partition, and creates a header portion of the partition of the hard disk 110.

Since the empty undefined region having the requested size is always aligned with the requested size by processing to be described later, it is possible to dispose a partition having the concerned size in this empty region.

When no empty region exists as a result of checking whether or not there is an empty undefined region having the requested size, the partition management execution unit 102 checks whether or not there is

an empty undefined region having a size 2m times (2, 4, 6, 8...) the requested size (S102).

As a result, when the empty undefined region having the size 2m times the requested size exists, the partition management execution unit 102 continues to divide the empty undefined region into halves till the empty undefined region has the requested size, and creates new regions (S103). And when a region having the requested size is created, the partition management execution unit 102 creates a partition in the region (S106). Meanwhile, the partition management execution unit 102 updates the partition management table 103 with other newly created region as empty undefined regions, and creates a header portion.

Fig. 5 is a view for explaining one example of the above processing. This drawing shows an example of the case where the creation of a partition having a size 2 is requested in a state where a sector number 8 has an empty undefined region having a size 8 (Fig. 5A).

Since the empty undefined region having the size 8, which starts from the sector number 8, is 4 times as large as the requested size 2, this region meets the condition of 2m times. First, this region is divided into halves. Then, two empty undefined regions having a size 4 each are created (Fig. 5B). And, an empty undefined region having a lower sector number is further divided into halves. Then, two empty undefined regions having a size 2 each are created (Fig. 5C).

Since these empty undefined regions have a size equal to the requested size, the partition is created in an empty undefined region having the size 2 and having a lower sector number. And, the partition management execution unit 102 updates the partition management table

103 with other newly created regions as empty undefined regions, and creates and updates a header (Fig. 5D).

When there is not an empty undefined region having a size 2m times as large as the requested size as a result of checking whether or not there is an empty undefined region 2m times as large, the partition management execution unit 102 checks whether or not the final undefined region is aligned with the requested size (S104).

As a result, when the final undefined region is aligned with the requested size, the partition management execution unit 102 creates the partition at the requested size from a starting position of the final undefined region (S106). And, the partition management execution unit 102 sets a remaining region as a final undefined region.

When the final undefined region is not aligned with the requested size, the partition management execution unit 102 creates an empty undefined region to a position where alignment is made (S105), and creates the partition at the requested size therein (S106).

Fig. 6 is a view for explaining one example of the above processing. This example is an example of the case where a partition with a size 1024 is requested when the final undefined region starts from a sector number 512 (Fig. 6A). Since the size 1024 is not aligned with the sector number 512 as a starting position of the final defined region, the partition management execution unit 102 sets a region from the sector number 512 to the sector number 1023 as an empty undefined region, and creates the partition with the size 1024 at the sector number 1024. Furthermore, the partition management execution unit 102 sets, as a final undefined region, a region of a sector number 2048 and thereafter which

region is a remaining region (Fig. 6B). And, the partition management execution unit 102 updates the partition management table 103, and creates and updates a header portion.

The partitions are created as described above, so that the partitions can be disposed in the positions aligned with the requested sizes.

The processing of deleting a partition will be explained with reference to a flowchart of Fig. 7.

The interface unit 101 of the partition management system 100 requests the partition management execution unit 102 to execute deletion of a partition upon receipt of a deletion request designating a specified partition from an operator, the program under execution, or the like.

The method of designating a partition to be deleted can be a method of designating an identifier such as a starting sector number, a partition number, or the like, of the partition. Naturally, the present invention is not limited to such a method, and the partition to be deleted can be specified by information to which the partition management execution unit 102 can refer and which permits designation of the partition to be deleted.

First, the partition management execution unit 102 deletes the designated partition, and sets a region thereof as an undefined region (S201).

Then, the partition management execution unit 102 judges whether or not the deleted partition is the last one of the defined partitions, that is, whether or not a region immediately after the deleted partition has been the final undefined region (S202).

As a result, when the deleted partition has been the last partition, the partition management execution unit 102 incorporates the undefined region caused by such deletion into the final undefined region (S203). Further, the partition management execution unit 102 judges whether or not a region immediately before the deleted partition has been an empty undefined region (S204), and when the region is the empty undefined region, the partition management execution unit 102 also incorporates the region into the final undefined region (S205).

On the other hand, when the deleted partition is not the last partition, the partition management execution unit 102 judges whether or not a region immediately before the region that has been set as an undefined region is an empty undefined region (S206). Then, when the region is an empty undefined region, the partition management execution unit 102 judges whether or not the region is aligned when the two undefined regions are combined (S207), and when the region can be aligned, the two undefined regions are combined and set as one undefined region (S208).

Further, the partition management execution unit 102 judges whether or not a region immediately after the empty undefined region caused by the deletion or the empty undefined region caused by such combination is an empty undefined region (S209). Then, when the region is an empty undefined region, the partition management execution unit 102 judges whether or not the region can be aligned when the two undefined regions are combined (S210), and when the region can be aligned, the two undefined regions are combined and set as one undefined region (S211).

Then, the partition management execution unit 102 reports the deletion of the partition to the requester through the interface unit 101. Further, the partition management execution unit 102 updates the header portion of each partition and the partition management table 103, and terminates the processing of deleting the partition.

Fig. 8 is a view for explaining one example of the above processing. When a partition D is deleted in Fig. 8A, a region immediately after the region from which the partition D is deleted is the final undefined region, so that the region from which the partition D has been deleted is added to the final undefined region as shown in Fig. 8B (S203).

When a partition D is deleted in Fig. 8C, the region from which the partition D is deleted is added to the final undefined region. In this case, a region immediately before that region is an undefined region F, so that the undefined region F is also added to the final undefined region as shown in Fig. 8D (S205).

When a partition D is deleted in Fig. 8E, if the total size of a region from which the partition D is deleted and an undefined region F immediately before that region can be aligned, these two partitions are combined to be set as one undefined region G as shown in Fig. 8F (S208). On the other hand, if the total size is cannot be aligned, two undefined regions that are an undefined region D as a result of deleting the partition D and an undefined region F are generated as shown in Fig. 8G.

When the partition size is limited to 2 to the n-th power, simulation results about utilization efficiencies of the hard disk will be shown with regard to a case (A) where the partition is disposed only at

the position aligned with the partition size and a case (B) where the partition is disposed at an arbitrary position below.

As a simulation method, 50 partitions having eight sizes of 8 megabytes, 16 megabytes, 32 megabytes, 64 megabytes, 128 megabytes, 256 megabytes, 512 megabytes and 1 gigabyte are pre-created on the hard disk at random, processings (1) to (3) shown below are iterated 1000 times while changing types of random numbers, and then, states of the hard disk are compared.

(1) Deleting one partition by use of random numbers;

(2) Selecting any one of the above-described eight sizes by use of the random numbers, and adding a partition of the size; and

(3) Iterating the processings of the above (1) and (2) 100 times.

As a result, the total numbers of sectors to the last partition (total number of sectors) and percentages of empty regions included in the total sectors to the last partition (empty/total) were as shown in Table 1 below.

[Table 1]

	A	B
Total Number of Sectors	35,287,040	33,820,672
Empty/Total	25.95%	21.78%

The above simulation results show that the present invention improves the utilization efficiency of the hard disk.

As described above, the utilization efficiency of the storage device can be improved according to the present invention.